CS-171 Wumpus World Final AI Report

- **Team name AlllllllByMyself**

Member #1 (name/id) Justin Wong/ 54158350 Member #2 (name/id or N/A) N/A

I. In about 1/2 page of text, describe what you did to make your Final AI agent "smart."

In the beginning my AI would randomly traverse the map and at any risk factor such as a stench or a bump I would turn and move back to a safe tile. My Agent would repeat this process until it randomly finds the gold which would set off a condition to pop off all my visited nodes in a stack until I returned to the origin. If the Agent didn't find the gold then I would set a condition where if it randomly visited the Origin over 8 times then it's time to climb because no gold was found.

I found that that process was very inefficient and wasted a lot of my moves on randomly revisiting the same tiles to only having a chance of finding gold which made my score very low. It was sufficient enough for my agent to obtain a score of over 100 but not enough to hit 200. In order to make my agent smart I knew I had to find a way to create a sort of memory bank or a way of processing tiles such that if I went back to the origin I would not revisit the tiles that didn't help lead me to my goal or in other words the shortest path back.

To do this I decided to change my AI so that it would implement a graph searching algorithm and the one I decided to use was DFS. DFS would make my agent smart because as I traverse the map my agent would push on the current tile to the stack if it were safe and pop it off if the tile wasn't so that if I were to follow a path back to the origin using my stack it would not contain moves where I went to tiles that weren't deemed safe and had to reverse. This way allowed me to have my agent memorize a clear path to the gold and to use that path back as I popped tiled off the stack to go back to the origin in a way more efficient manner versus following the exact way and the amount of turns the agent took to get to the AI. In the end this saved me a lot of moves but at the same time DFS was not the only thing I had to consider.

In order for me to use DFS efficiently I had to also record lists of explored nodes and predicted moves around my current tile for my agent. This way I would know when to pop off a node from my visited stack and turn back because if my agent made a prediction about the tiles it can go to that's around my agent, if some of those tiles have already been explored then I would go and pick the unexplored ones and have my agent move to those. If in the end my agent was surrounded by explored tiles meaning my agent has no other paths to new nodes or tiles then just like DFS my agent would pop off the current location off the stack and move to the location on the top of the stack. At this new location my agent would repeat the process of looking for unexplored tiles/nodes and go to those until my agent has no more possible moves or is back at the origin then it would mean it tried every move possible to get to the gold but the gold was probably trapped behind too many risk factors. If my agent however obtained the gold then I would stop using the prediction process and just constantly pop off nodes off my stack and follow those tiles home to the origin where I climbed.

This process in turn made my Agent smart as it simulated a memory bank that the agent would use to find the best path home and not re-traverse spots that it already visited.

II. In about 1/4 page of text, describe problems you encountered and how you solved them.

Problems that I mostly encountered had to do with my conditions, segmentation faults when I accidentally tried to pop off empty stacks if I was at the origin, and storing valid values in my predictions list for the moves around me so that I could choose from one. Either my conditions or arithmetic made it so that I didn't store the correct values in my possible moves list or my program would crash if I got stuck in a spot and it would try to select a random move near me that it couldn't make a prediction list for. In order to fix all these problems I had to go through my code multiple times run it over and over again print out my lists and stacks and adjust my arithmetic until the values were correct.

III. In about 1/4 page of text, provide suggestions for improving this project

Maybe have a short explanation of how the shell is organized in the beginning because that took me a good amount of time for me to be able to understand how the shell processed moves for each iteration.